

Programação Orientada a Objetos

Aula Teórica: Turma B e Turma C
16/12/2021

Orientação a Objetos (continuação)
Abstração, Classes e Objetos

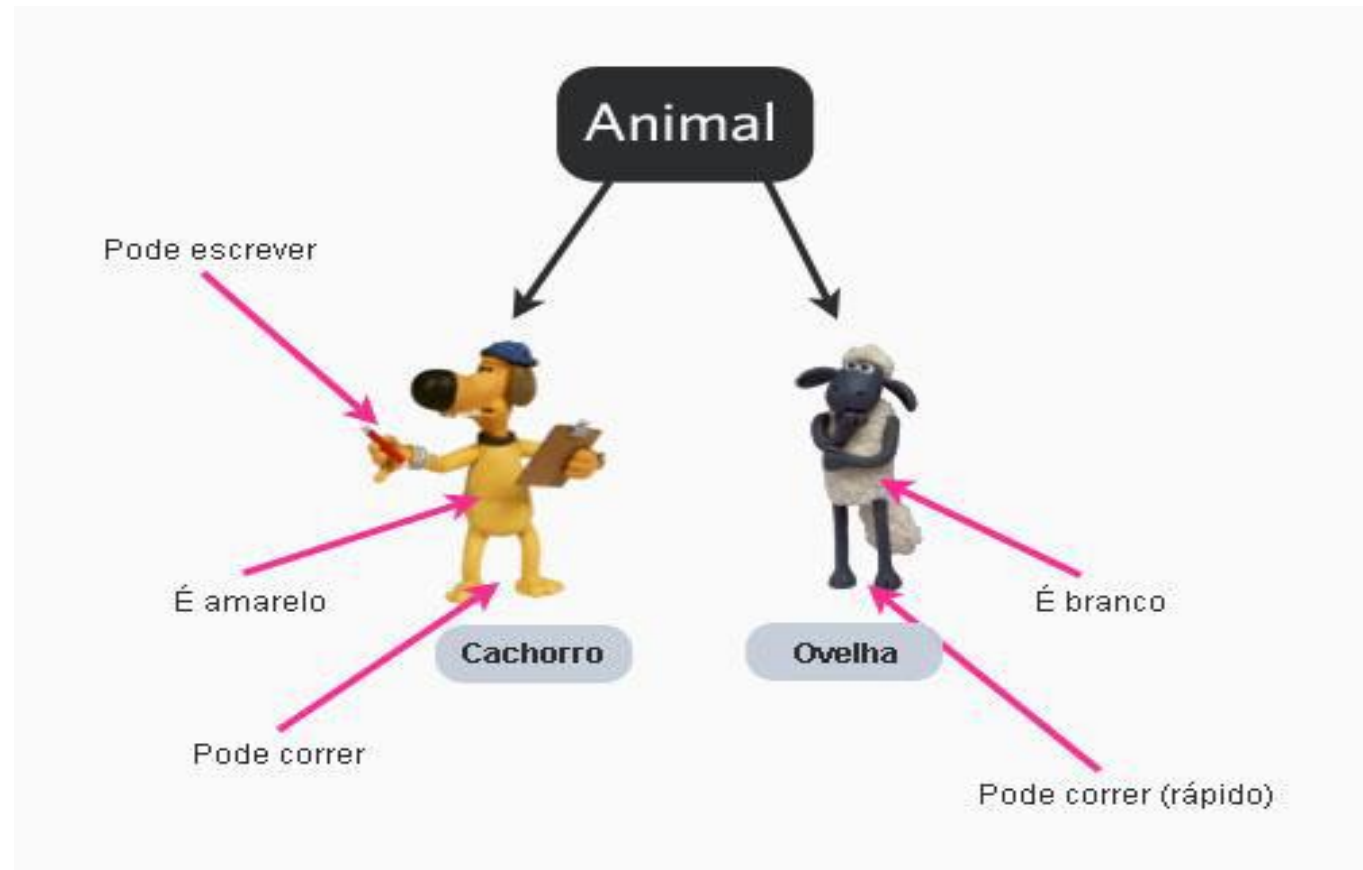
Profa. Nadia Félix
nadia.felix@ufg.br

Profa. Dirson Santos de Campos
dirson_campos@ufg.br

Programação Orientada a Objetos

ABSTRAÇÃO

Classes e Objetos



Programação Orientada a Objetos

Abstração

- Habilidade de se concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais;
- Na Orientação a Objetos, uma **classe** é uma abstração de entidades existentes no domínio do sistema de software;

Programação Orientada a Objetos

Classificando



O que identificamos nessa foto ?

Alunos, Carteiras, Cadeiras, Quadros, Professores, Pessoas

Programação Orientada a Objetos

Classificando

Todos os Alunos são iguais? **Não**

Cada aluno é um objeto diferente do cenário mas todos são considerados/classificados como **Alunos**.

Logo, **Aluno** é uma **Classe**.

Características de um Aluno (atributos):

Nome

Cor do cabelo

Idade

Sexo

Altura

Peso



X



Programação Orientada a Objetos

Classificando

O que identificamos nessa foto ?

- calças
- casacos
- camisas
- saias
- bolsas
- sapatos
- vestidos



Programação Orientada a Objetos

Classificando

Todas as Camisas são iguais?

Cada camisa é um objeto diferente do cenário mas todos são considerados/classificados como **Camisa**.

Logo, **Camisa** é uma **Classe**.

Características de uma camisa (atributos):

Cor

Tamanho

Modelo

Marca

Objetos diferentes da classe

Camisa

Instâncias

Camisa 1



Camisa 2



Programação Orientada a Objetos

Classe

- A classe é a unidade básica de trabalho em um programa orientado a objetos;
- Representa um modelo abstrato a partir do qual são criadas instâncias (objetos);
- É uma coleção de dados e operações que manipulam tais dados;
- Consiste de dois tipos de elementos: **atributos** (dados/características) e **métodos** (operações/comportamentos).

Programação Orientada a Objetos

Classe **Pessoa**

•Exemplos de Características / Atributos:

- Nome
- Idade
- Sexo
- Nacionalidade
- Raça
- etc



Programação Orientada a Objetos

Classe **Pessoa**

.Exemplos de Comportamentos / Métodos:

- Respirar
- Dormir
- Andar
- Comer
- etc



Programação Orientada a Objetos

Objeto

- Objeto é uma **instância** da classe;
- É uma representação real da classe que ocupa espaço na memória e consome recursos do computador;
- A classe é a descrição de um tipo de objeto;
- Uma classe pode ser considerada uma “fábrica” para criação de objetos;

Programação Orientada a Objetos

Atributos & Métodos

–A classe é uma coleção de dados(atributos) e operações(métodos) que manipulam tais dados

Atributos

- São os dados (simples ou compostos) que caracterizam os objetos daquela classe;
- São armazenadas em variáveis;
- Constituem o **estado** do objeto.

Métodos

- São as operações (procedimentos ou funções) que manipulam os dados;

Programação Orientada a Objetos

Exercício

–Cite 3 atributos e 3 métodos de cães



–Cite 3 atributos e 3 métodos de motos



Programação Orientada a Objetos

Resposta do Exercício (Cães)

-Cite 3 atributos e 3 métodos de Cães

•Atributos

-Raça

-Nome

-Peso

•Métodos

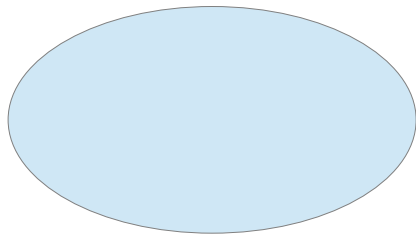
-Latir

-Andar

-Comer

-Crie 3 instâncias/objetos de Cães

(Cada instância tem seu próprio estado)



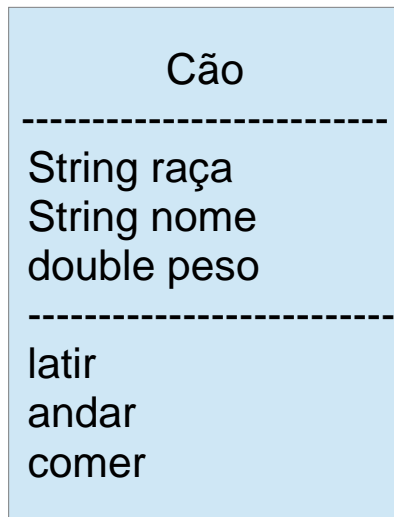
Estado



Programação Orientada a Objetos

Diferenciando Classes de Objetos

Classe



Objetos



Raça: Labrador
Nome: Marley
Peso: 7 Kg



Raça: Poodle
Nome: Mellow
Peso: 3 Kg



Raça: Beagle
Nome: Safira
Peso: 10 Kg

Cada objeto tem seu conjunto de atributos – representam o **estado** do objeto;

Programação Orientada a Objetos

Resposta Exercício (motos)

–Cite 3 atributos e 3 métodos de motos

•Atributos

–Ano

–Modelo

–Potência

•Métodos

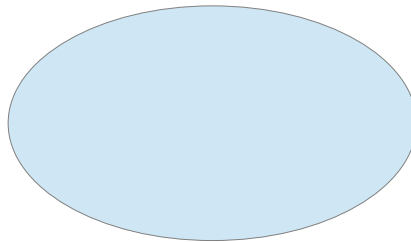
–Frear

–Acelerar

–Abastecer

–Crie 3 instâncias/objetos de Moto

(Cada instância tem seu próprio estado)



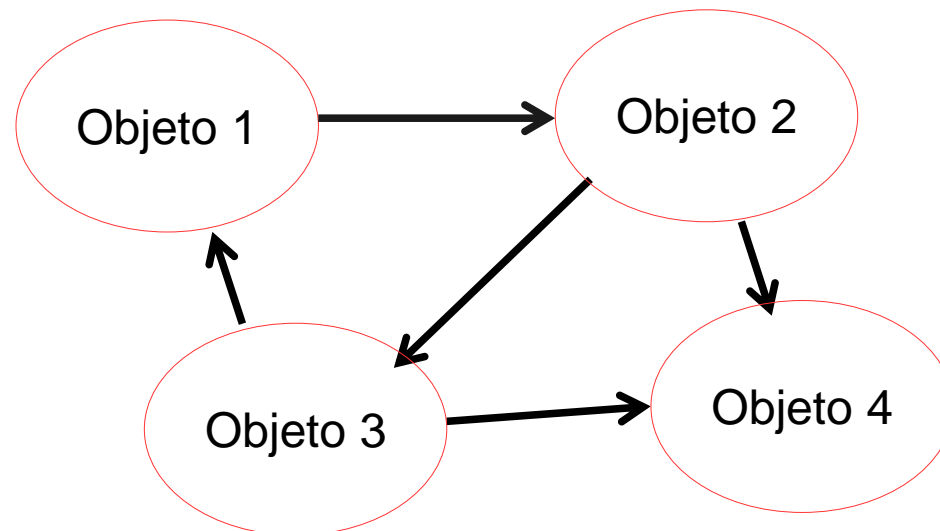
Estado



Programação Orientada a Objetos

Orientação a Objetos

–A **orientação a objetos** é um modelo de análise, projeto e programação de sistemas de software baseado na **composição** e **interação** entre diversas unidades de software chamadas de **objetos**.



Programação Orientada a Objetos

Modelagem com UML

–A Unified Modelling Language (UML) é uma linguagem ou notação de diagramas para especificar, **visualizar** e **documentar** modelos de 'software' orientados por objetos.

–A UML é composta por muitos elementos de modelo que representam as diferentes partes de um sistema de software. Os elementos UML são usados para criar **diagramas**, que representam um determinada parte, ou um ponto de vista do sistema.

– A UML é uma linguagem que está em uso, por este motivo, com o tempo, é lançado melhorias técnicas que geral novas versões. A UML é um padrão do OMG (Object Management Group) que é é uma organização internacional que aprova padrões abertos para aplicações orientadas a objetos.



Programação Orientada a Objetos

Alguns Diagramas da UML

- **Diagrama de Caso de Uso** mostra atores (pessoas ou outros usuários do sistema), casos de uso (os cenários onde eles usam o sistema), e seus relacionamentos
- **Diagrama de Classe** mostra classes e os relacionamentos entre elas
- **Diagrama de Sequência** mostra objetos e uma sequência das chamadas do método feitas para outros objetos.
- **Diagrama de Colaboração** mostra objetos e seus relacionamentos, colocando ênfase nos objetos que participam na troca de mensagens
- **Diagrama de Estado** mostra estados, mudanças de estado e eventos num objeto ou uma parte do sistema
- **Diagrama de Atividade** mostra atividades e as mudanças de uma atividade para outra com os eventos ocorridos em alguma parte do sistema
- ...

Programação Orientada a Objetos

Criando uma Classe - Modelo

–Pessoa

•O que uma pessoa tem? **(Atributos / Características / Dados)**

–Nome

–Idade

–CPF

–Identidade

–Sexo

•O que uma pessoa faz? **(Métodos / Operações / Comportamentos)**

–Acordar

–Andar

–Comer

–...

Programação Orientada a Objetos

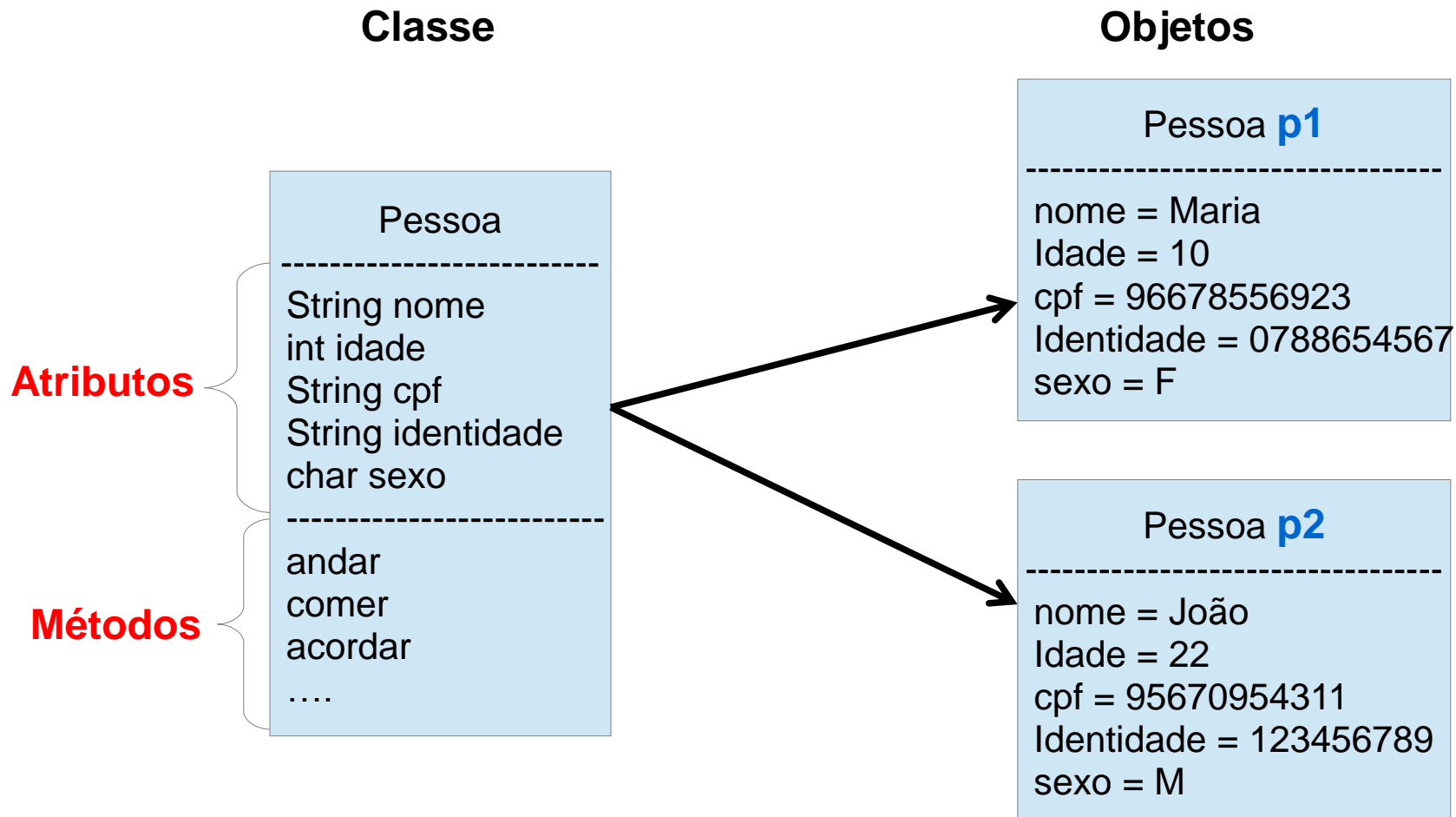
Abstraindo Contextos

.Exemplos:

Contexto	Dados (atributos)	Operações (métodos)
Funcionário de uma Empresa	Nome, cargo, salario, horasExtras	CalculaSalario, aumentaSalario
Paciente de uma clínica	Nome, sexo, idade, altura, peso, historicoDeConsultas	VerificaObesidade, adicionalInformacoesHistorico
Contato Comercial	Nome, telefone, cargo, empresa	MostraTelefone, trabalhaEmEmpresa

Programação Orientada a Objetos

Representação UML



Programação Orientada a Objetos

Criando uma Classe em Java

```
class Pessoa {
```

```
    String nome;
```

```
    int idade;
```

```
    String cpf;
```

```
    String identidade;
```

```
    char sexo;
```

```
    ...
```

```
    public void andar() {
```

```
        .....
```

```
    }
```

```
    public void comer() {
```

```
        .....
```

```
    }
```

```
    public void acordar() {
```

```
        ..... } }
```

Declarando os atributos da classe



Declarando os métodos da classe



Programação Orientada a Objetos

Criando um Objeto em Java

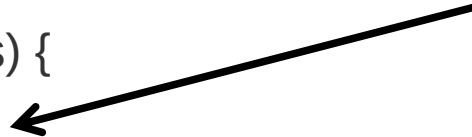
- Precisa ser **identificado** enquanto entidade do código (uma variável, um elemento em uma lista, uma posição de um vetor, etc.);
- O operador **new**...
 - Aloca memória para o novo objeto (a quantidade necessária é obtida a partir da definição da classe);
 - Chama o construtor da classe para inicializar o estado do novo objeto;
 - Retorna uma referência (um endereço de memória para o objeto recém criado);

Programação Orientada a Objetos

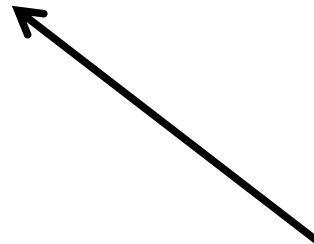
Usando a Classe e criando Objetos em Java

```
class Programa {  
    public static void main (String[] args) {  
  
        Pessoa cliente;  
        cliente = new Pessoa();  
    }  
}
```

Define e cria a variável cliente



Aloca Memória e devolve um valor de referência para o Objeto do tipo Pessoa

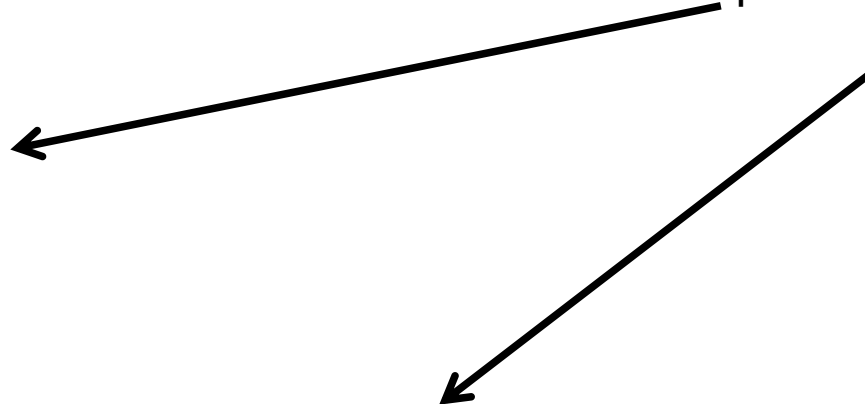


Programação Orientada a Objetos

Acessando o objeto e alterando seus dados

```
class Programa {  
    public static void main (String[] args) {  
        Pessoa cliente;  
        cliente = new Pessoa();  
  
        cliente.nome = "Maria";  
  
        System.out.println("Nome do Cliente = " + cliente.nome);  
    }  
}
```

Usar o ponto "." para acessar os dados



Programação Orientada a Objetos

Exercício

- Criar uma classe Pessoa com os atributos: nome, idade, sexo e telefone; e o método que retorna os dados de uma Pessoa concatenados ('+').
- Criar um programa que use a classe Pessoa para instanciar três objetos diferentes e mostrar seus dados.

Programação Orientada a Objetos

Resposta do Exercício Criando um objeto em Java

Classe

```
class Pessoa {  
    String nome;  
    int idade; long cpf;  
    long identidade;  
    char sexo;  
    void andar() {...}  
    void comer() {...}  
    void acordar() {...}  
}
```

Objetos

```
Pessoa p1 = new Pessoa();  
p1.nome = "Maria";  
p1.idade = 10;  
p1.cpf = 96678556923; identidade = 0788654567;  
p1.sexo = 'F';  
p1.acordar();
```

new

new

```
Pessoa p2 = new Pessoa();  
p2.nome = "João";  
p2.idade = 22;  
p2.cpf = 95670954311;  
p2.identidade = 123456789;  
p2.sexo = 'M';  
p2.andar();
```

Programação Orientada a Objetos

Resposta do Exercício

Usando a Classe Pessoa e criando Objetos

```
class Programa {  
public static void main (String[] args) {  
Pessoa p1 = new Pessoa();  
p1.nome = "Maria";  
p1.idade = 10;  
p1.cpf = 96678556923;  
p1.identidade = 0788654567;  
p1.sexo = 'F';
```

```
Pessoa p2 = new Pessoa();  
p2.nome = "João";  
p2.idade = 22;  
p2.cpf = 95670954311;  
    p2.identidade = 123456789;  
    p2.sexo = 'M';
```

```
System.out.println("Criei duas instâncias de Pessoa: ");  
System.out.println("Primeira – Nome : " + p1.nome + " - Idade : " + p1.idade + " - Sexo : " + p1.sexo );  
System.out.println("Segunda – Nome : " + p2.nome + " - Idade : " + p2.idade + " - Sexo : " + p2.sexo );
```

Programação Orientada a Objetos

Referência **null**

- Valor default para inicialização de atributos de tipos não primitivos;
- Indica que determinado atributo/variável de referência não se refere a nenhum objeto;

Programação Orientada a Objetos

Abstraindo no Domínio de um Banco

.Informações Relevantes

-Cliente

-Conta

.Conta corrente

.Conta poupança

.Conta salário

-Investimento

-Contrato

-Convênio

-Boleto Bancário

-Cheque

-Funcionário

.Gerente

.Caixa

-Agência

-Pagamento

-Financiamento

-Etc.

Programação Orientada a Objetos

Criando um tipo de dado – Uma Classe

–Modelo

•Conta

–O que uma conta tem? **(Atributos)**

•Número da conta

•Nome do dono

•Tipo da conta

•Saldo

•Limite

–O que uma conta faz? **(Métodos)**

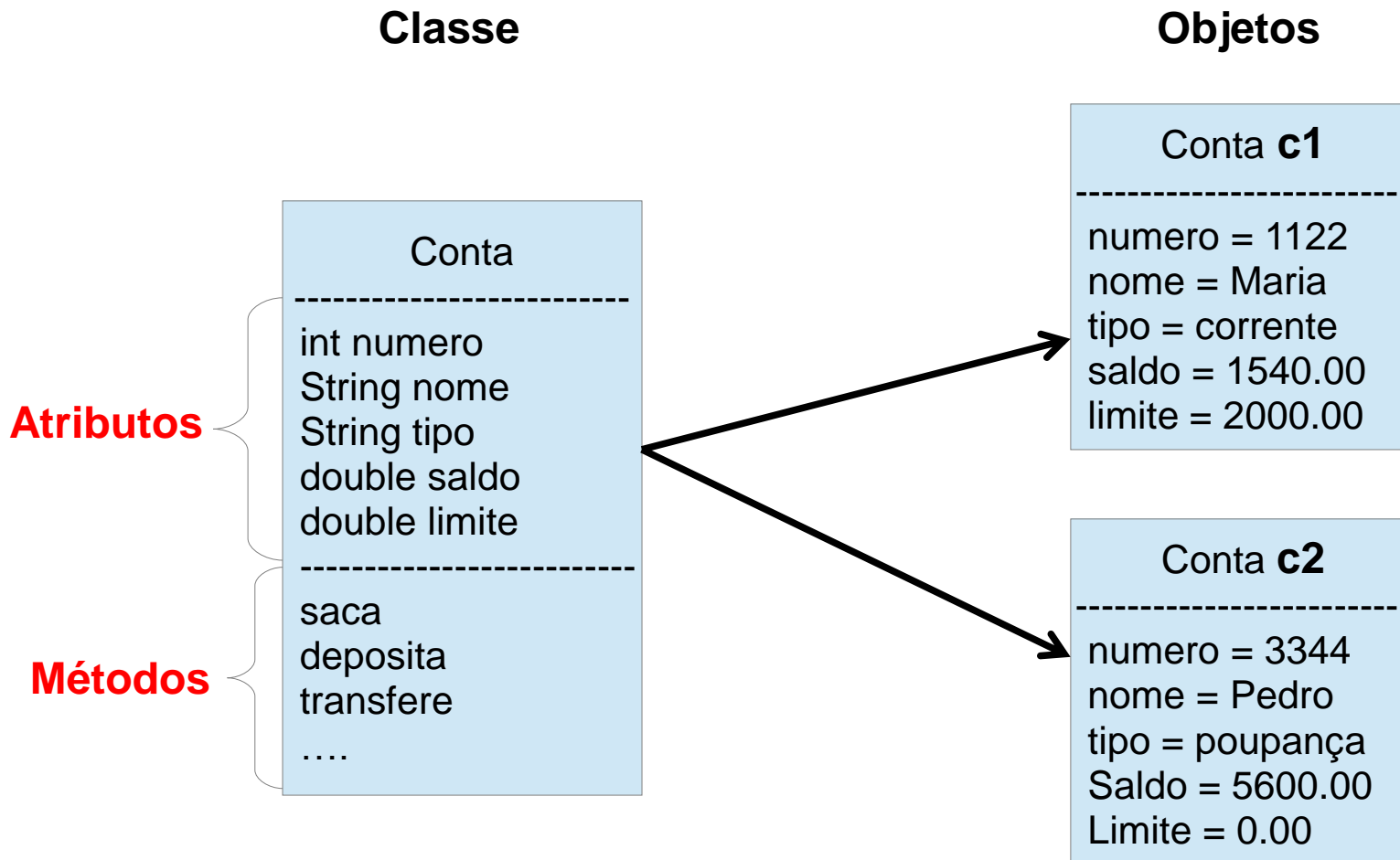
•Saca um valor x

•Deposita um valor x

•Transfere um valor x para outra conta c2

Programação Orientada a Objetos

Representação UML



Programação Orientada a Objetos

Classe em Java

Conta

.Atributos : O que toda conta deve ter?

-

```
.class Conta {  
int numero;  
String nome;  
String tipo;  
double saldo;  
double limite;  
...  
.}
```

Programação Orientada a Objetos

Classe em Java

Conta

.Métodos : O que toda conta pode fazer?

-

```
.class Conta {
```

```
...
```

```
// definição dos atributos
```

```
...
```

```
void saca(double valor) {
```

```
    saldo = saldo - valor;
```

```
}
```

```
.}
```

Programação Orientada a Objetos

Classe em Java

```
class Conta {  
    // definição dos atributos  
    int numero;  
    String nome;  
    String tipo; saldo;  
    double limite;  
  
    //definição dos métodos  
    void saca(double valor) {  
        saldo = saldo - valor;  
    }  
    void deposita(double valor) {  
        saldo = saldo + valor;  
    }  
    void transfere(double valor, Conta c2) {  
        saldo = saldo - valor;  
        c2.saldo = c2.saldo + valor;  
    }  
}
```

Programação Orientada a Objetos

Usando a Classe e criando Objetos

```
class Programa {  
    public static void main (String[] args) {  
  
        Conta minhaConta;  
        minhaConta = new Conta();  
    }  
}
```

Define e cria a variável minhaConta



Aloca Memória e devolve um valor de referência para o Objeto do tipo Conta

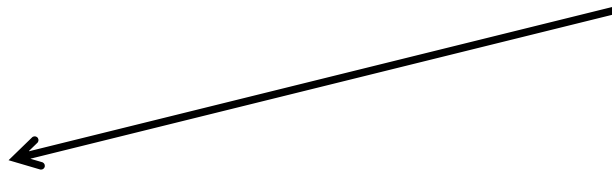


Programação Orientada a Objetos

Acessando o objeto e alterando seus dados

```
class Programa {  
    public static void main (String[] args) {  
        Conta minhaConta;  
        minhaConta = new Conta();  
  
        minhaConta.saldo = 200.00;  
        minhaConta.nome = "Maria";  
  
        System.out.println("Saldo Atual = " + minhaConta.saldo);  
    }  
}
```

Usar o ponto "." para acessar os dados



Programação Orientada a Objetos

Inserindo Métodos (operações)

–Método que saca uma determinada quantidade e não retorna nenhum valor, apenas atualiza o saldo.

```
class Conta {
```

```
...
```

```
double limite;
```

```
retorno
```

```
void saca(double valor) {
```

```
    saldo = saldo - valor;
```

```
}
```

```
}
```

parâmetros

void – indica que o método não retorna valor

Programação Orientada a Objetos

Inserindo Métodos (operações)

–Método que deposita uma determinada quantidade e não retorna nenhum valor, apenas atualiza o saldo.

```
class Conta {
```

```
// atributos e outros métodos
```

```
...
```

```
void deposita(double valor) {
```

```
    saldo += valor;
```

```
}
```

```
}
```


Programação Orientada a Objetos

Assinatura de Métodos (operações)

–Identifica o método como único

•Nome do método

•Lista de parâmetros do método

```
void deposita(double valor) {  
    saldo += valor;  
}
```

```
void deposita(double valor, int quantidade) {  
    saldo += valor * quantidade;  
}
```

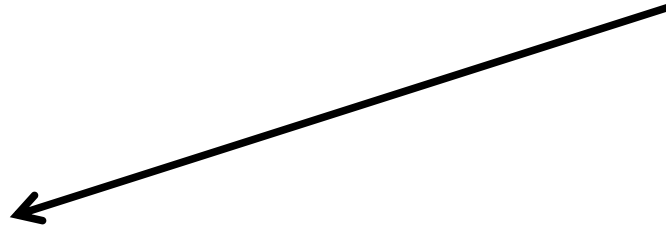
Métodos com mesmo nome
E assinaturas diferentes
Logo, **métodos diferentes**

Programação Orientada a Objetos

Testando a Classe Conta

```
class Programa {  
    public static void main (String[] args) {  
        Conta minhaConta;  
        minhaConta = new Conta();  
  
        minhaConta.saldo = 200.00;  
        minhaConta.nome = "Maria";  
  
        minhaConta.deposita(450.00);  
        MinhaConta.saca(100.00)  
  
        System.out.println("Saldo Atual = " + minhaConta.saldo);  
    }  
}
```

Invocando métodos



Programação Orientada a Objetos

Métodos com Retorno

```
class Conta {
```

```
...
```

```
double limite;
```

Retorna um boolean

```
.boolean saca(double valor) {
```

```
if (this.saldo < valor) {
```

```
return false;
```

```
}
```

```
eles {
```

```
this.saldo = this.saldo - valor;
```

```
return true;
```

```
}
```

```
}
```

Todo método com retorno de valores deve terminar com um **return**

```
class Programa {
```

```
public static void main (String[] args) {
```

```
Conta minhaConta = new Conta();
```

```
minhaConta.saldo = 200.00;
```

```
minhaConta.dono = "Maria";
```

```
if (minhaConta.saca(50.00)) {
```

```
System.out.println("Conseguiu Sacar");
```

```
}
```

```
eles {
```

```
System.out.println("Não conseguiu sacar");
```

```
}
```

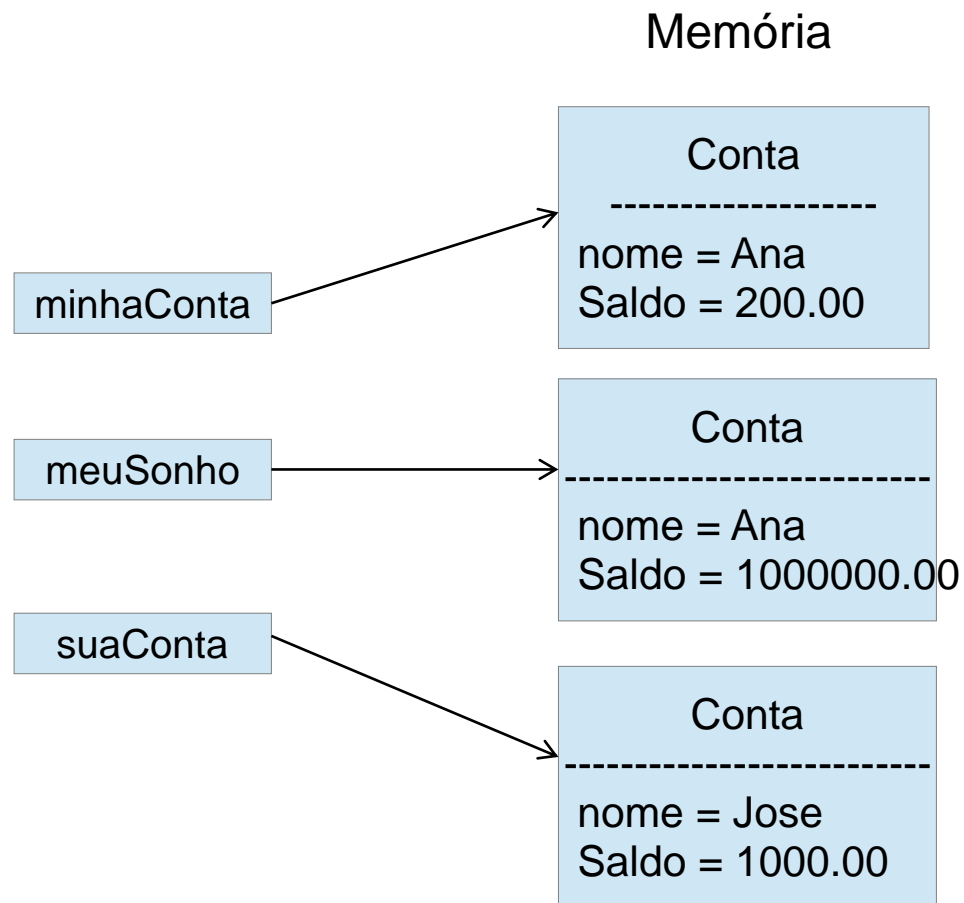
```
}
```

```
}
```

Programação Orientada a Objetos

Criando vários Objetos em Memória – várias Contas

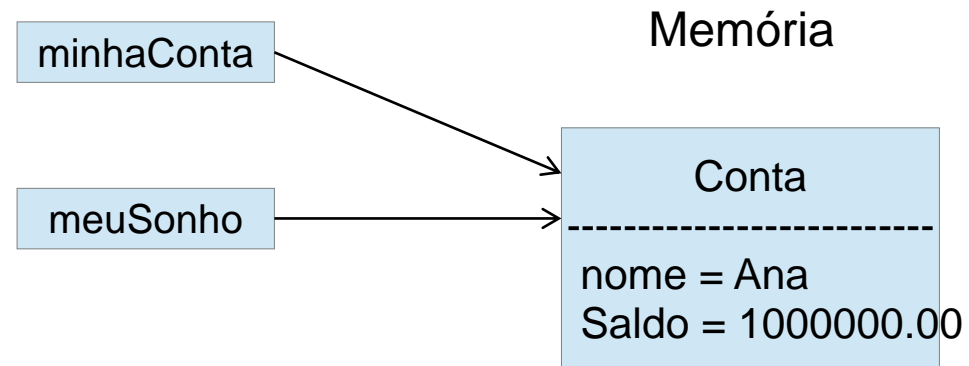
```
class Programa {  
    public static void main (String[] args) {  
        Conta minhaConta = new Conta();  
        minhaConta.saldo = 200.00;  
        minhaConta.nome = "Ana";  
  
        Conta meuSonho = new Conta();  
        meuSonho.saldo = 1000000.00;  
        meuSonho.nome = "Ana";  
  
        Conta suaConta = new Conta();  
        suaConta.saldo = 1000.00;  
        suaConta.nome = "José";  
    }  
}
```



Programação Orientada a Objetos

Atribuindo variáveis à mesma referência

```
class Programa {  
  
    public static void main (String[] args) {  
  
        Conta minhaConta = new Conta();  
        minhaConta.saldo = 200.00;  
        minhaConta.nome = "Ana";  
  
        Conta meuSonho = new Conta();  
        meuSonho.saldo = 1000000.00;  
        meuSonho.nome = "Ana";  
  
        minhaConta = meuSonho;  
        System.out.println(minhaConta.saldo);  
        System.out.println(meuSonho.saldo);  
    }  
}
```



Programação Orientada a Objetos

Método Transfere para outra Conta

```
class Conta {  
    ...  
    // atributos e métodos;  
  
    void transferePara(Conta destino, double valor) {  
        this.saldo = this.saldo - valor;  
        destino.saldo = destino.saldo + valor;  
    }  
}
```

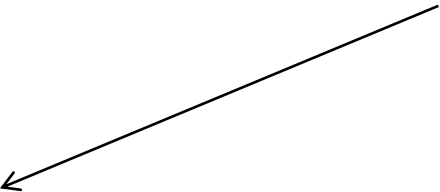
```
class Programa {  
    public static void main (String[] args) {  
        Conta minhaConta = new Conta();  
        minhaConta.saldo = 1000.00;  
  
        Conta contaDestino = new Conta();  
        contaDestino.saldo = 200.00;  
  
        minhaConta.transferePara(contaDestino,100.00);  
        System.out.println(minhaConta.saldo);  
        System.out.println(contaDestino.saldo);  
    }  
}
```

Programação Orientada a Objetos

Trabalhando com valores *default*

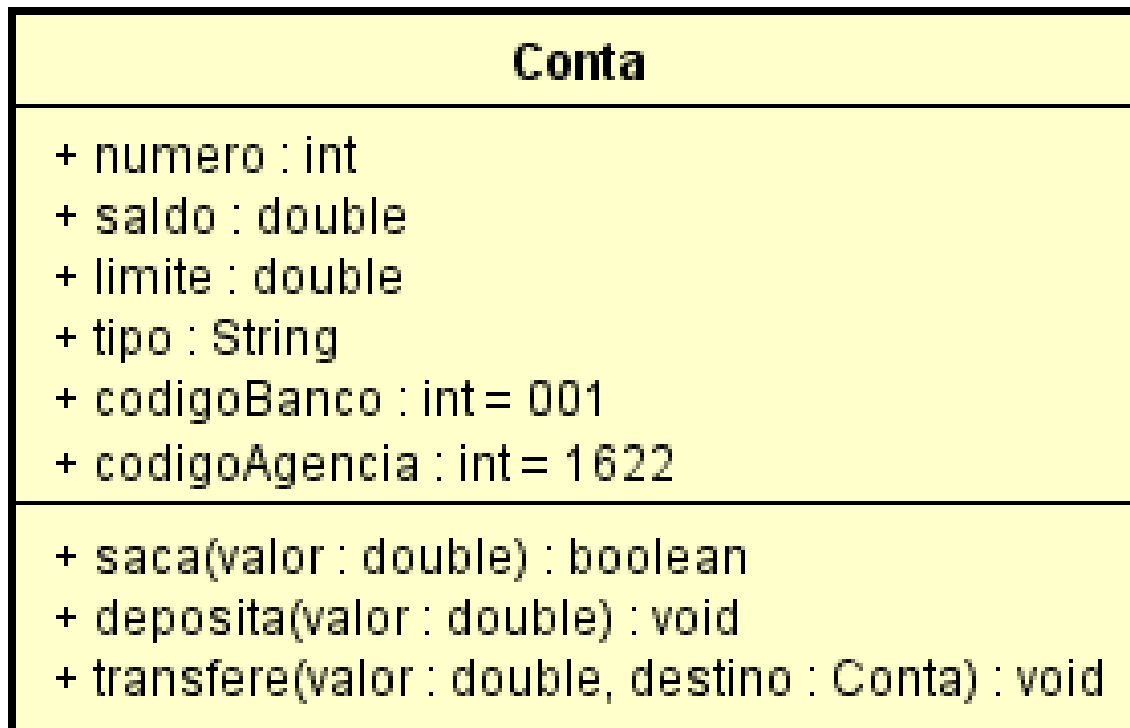
```
class Conta {  
    int numero;  
    String nome;  
    String tipo;  
    double saldo;  
    double limite;  
    int codigoBanco = 001;  
    int codigoAgencia = 1622;  
    ...  
}
```

Quando não for informados o Banco e a Agência, as contas serão criadas para o Banco 0001 e Agência 1622



Programação Orientada a Objetos

Modelo da Conta em UML



Programação Orientada a Objetos

Sobrecarga de Métodos

- Métodos de mesmo nome com assinaturas diferentes (tipos de argumentos/parâmetros diferentes);
- Valor de retorno não é considerado parte da assinatura;
- Métodos sobrecarregados são resolvidos em tempo de compilação;

```
public class Hora {  
  
    private int hora;  
    private int minuto;  
    private int segundo;  
  
    public void ajustarHorario(int hora, int minuto, int segundo);  
  
    {  
        public void ajustarHorario(int hora, int minuto);  
        public void ajustarHorario(int minuto, int segundo);  
  
        public void ajustarHorario(int hora);  
    }  
}
```

Estes dois métodos não tem assinaturas diferentes isso não é permitido

Programação Orientada a Objetos

Métodos Construtores

- Chamado automaticamente pelo ambiente de execução quando um objeto é criado - **NEW**;
- **SEMPRE** tem o mesmo nome da classe e pode opcionalmente receber parâmetros;
- Nunca pode ter um tipo de retorno (isso é implícito pois ele retorna uma referência para um objeto da classe em questão);

Programação Orientada a Objetos

Métodos Construtores

```
class Conta {  
    int numero;  
    double saldo;  
    double limite;  
  
    // construtor  
    Conta() {  
        System.out.println("Construindo uma conta.");  
    }  
    // ..  
}
```

Toda vez que fizer **new** Conta(), vai mostrar essa mensagem

Construtor Padrão é criado pelo Java com o corpo vazio e executado sempre que não houver outro declarado.

Toda classe tem um construtor padrão

Programação Orientada a Objetos

Métodos Construtores

- Só é utilizado na criação do objeto (**new**)
- Podem ter vários construtores com assinaturas diferentes
- São usados principalmente para inicialização de atributos

Programação Orientada a Objetos

Métodos Construtores

```
class Conta {  
    int numero;  
    double saldo;  
    double limite;  
  
    // construtor  
    Conta(int numero, double limite) {  
        this.numero = numero;  
        this.limite = limite;  
    }  
    // ..  
}
```

```
class Programa {  
    ...  
    ...  
    Conta minhaConta = new Conta(1122,100.00);  
    ...  
  
    //...  
}
```

Programação Orientada a Objetos

Referência **this**

• Todos os métodos de instância recebem um parâmetro implícito chamado `this` que pode ser utilizado dentro do método de instância para se referir ao objeto corrente;

• É normalmente utilizado para diferenciar os atributos do objeto das demais variáveis.

```
class Conta {  
    int numero;  
    double saldo;  
    double limite;
```

```
// construtor
```

```
    Conta(int numero, double limite) {  
        this.numero = numero;  
        this.limite = limite;  
    }
```

```
}
```

```
class Principal {
```

```
    public static void main (String[] args) {  
        Conta minhaConta = new Conta(1234 , 100.00);
```

Programação Orientada a Objetos

Método **finalize**

- Java não implementa o conceito de “destrutores” (métodos invocados para destruir a instância e liberar seus recursos);
 - Para isso há o conceito de “**coletor de lixo**” (garbage collector)
- Contudo, há situações onde é interessante que ao destruir o objeto, outros recursos, que não a memória, venham a ser liberados.
- Neste caso pode-se definir um método de nome **finalize** e retorno **void** que será chamado antes do coletor de lixo eliminar o objeto;

Programação Orientada a Objetos

Exercícios de Fixação (teóricos)

- 1) Quais são os membros de uma classe?
- 2) O que define o estado de um objeto?
- 3) Como são chamadas as características de um objeto?
- 4) Como são chamadas as operações que um objeto pode fazer?
- 5) O que são os construtores?
- 6) O que é a referência *this*?
- 7) O que é a sobrecarga de métodos?

Programação Orientada a Objetos

Exercícios de Fixação (práticos)

1)

a) Crie uma classe que represente os vários cursos de uma universidade com 4 atributos.

.b) Crie uma classe Principal que instância 4 cursos diferentes da universidade e mostre os dados de cada curso.

.Laboratório:

-1) A classe principal deve solicitar ao usuário os dados dos 4 cursos, instanciá-los e no final mostrar os dados de cada curso.

-2) Repetir o procedimento acima mas guardar os objetos em um vetor e depois ler o vetor e mostrar os dados dos objetos.